Харківський університет радіоелектроніки
Факультет комп'ютерних наук
Кафедра програмної інженерії

ЗВІТ

до практичного заняття

з дисципліни "Аналіз та рефакторинг коду"

на тему: "Refactoring Methods"

Виконав ст. гр ПЗПІ-22-2
Терновий Данііл Павлович

Перевірив доцент кафедри ПІ
Лещинський Володимир Олександрович

Харків 2024

МЕТА РОБОТИ

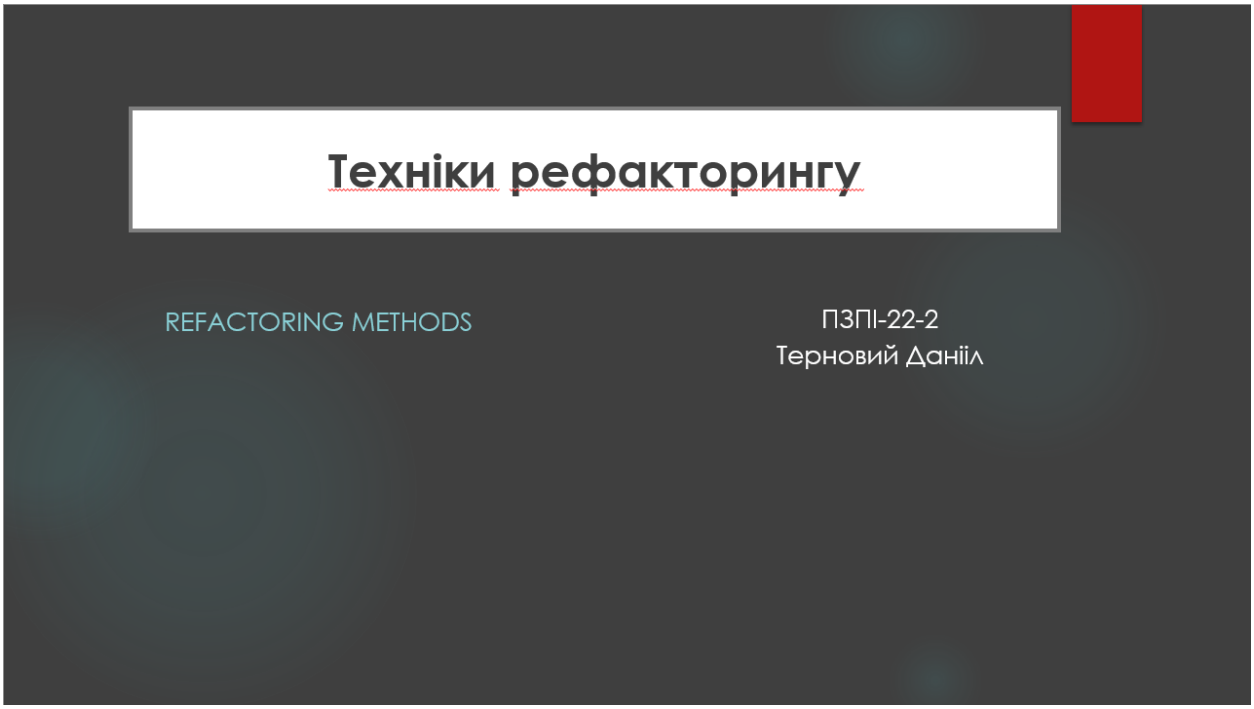
Навчитись основним методам рефакторингу коду на основі реальних прикладів з власних програмних проєктів та навчитись ідентифікувати проблеми в коді та використовувати відповідні методи рефакторингу для покращення його якості.

ВИСНОВКИ

У ході виконання завдання я опанував основні методи рефакторингу на прикладах з власних проєктів, та навчився розпізнавати проблемні місця в коді та ефективно застосовувати відповідні техніки для покращення його якості.

ДОДАТОК А

Слайди до презентації



Change reference to value

ДО

```
public void CalculateTotal(ref int total)
{
    total += 10;
}
```

Після

```
public int CalculateTotal(int total)
{
    int localTotal = total;
    localTotal += 10;
    return localTotal;
}

int total = 5;
total = CalculateTotal(total);
```

Change value to reference

ЦЕЙ ПРИЙОМ РЕФАКТОРИНГУ ПЕРЕДБАЧАЄ ЗМІНУ ПАРАМЕТРА-ЗНАЧЕННЯ НА ПАРАМЕТР-ПОСИЛАННЯ. ЦЕ ДОЗВОЛЯЄ МЕТОДУ ЗМІНЮВАТИ ПОЧАТКОВЕ ЗНАЧЕННЯ, ЩО МОЖЕ БУТИ КОРИСНИМ ДЛЯ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ПРИ РОБОТІ З ВЕЛИКИМИ ОБ'ЄКТАМИ.

Change value to reference

До

```
1  public void UpdatePrice(decimal price)
2  {
3      price += 5;
4  }
```

ПІСЛЯ

```
1  public void UpdatePrice(ref decimal price)
2  {
3      price += 5;
4  }
5
6  decimal price = 100;
7  UpdatePrice(ref price);
```

Encapsulate collection

ЦЕЙ ПРИЙОМ РЕФАКТОРИНГУ ПЕРЕДБАЧАЄ ПРИХОВУВАННЯ ПРЯМОГО ДОСТУПУ ДО КОЛЕКЦІЇ, НАДАЮЧИ КОНТРОЛЬОВАНИЙ ДОСТУП ЧЕРЕЗ МЕТОДИ. ЦЕ ДОПОМАГАЄ ЗАПОБІГТИ НЕСПОДІВАНІЙ МОДИФІКАЦІЇ КОЛЕКЦІЇ ЗОВНІШНІМ КОДОМ, РОБЛЯЧИ СТАН ОБ'ЄКТА БЕЗПЕЧНІШИМ.

Encapsulate collection

До

```
public class Order
{
    public List<string> Items { get; set; } = new List<string>();
}

var order = new Order();
order.Items.Add("Item1");
order.Items.Add("Item2");
```

ПІСЛЯ

```
public class Order
{
    private readonly List<string> _items = new List<string>();

    public IReadOnlyList<string> Items => _items.AsReadOnly();

    public void AddItem(string item)
    {
        _items.Add(item);
    }

    public void RemoveItem(string item)
    {
        _items.Remove(item);
    }
}

var order = new Order();
order.AddItem("Item1");
order.AddItem("Item2");
foreach (var item in order.Items)
{
    Console.WriteLine(item);
}
```

Висновок

РЕФАКТОРИНГ НЕОБХІДНИЙ ДЛЯ ТОГО, ЩОБ КОД ЗАЛИШАВСЯ ЧИСТИМ І ЗРУЧНИМ ДЛЯ ОБСЛУГОВУВАННЯ. ЗАСТОСОВУЮЧИ ЦІ МЕТОДИ, РОЗРОБНИКИ МОЖУТЬ ПОКРАЩИТИ ЯКІСТЬ КОДУ ТА ЗАПОБІГТИ ПОТЕНЦІЙНИМ ПРОБЛЕМАМ У МАЙБУТНІЙ РОЗРОБЦІ.